

Travlr Full Stack Web Application

# **Project Software Design Document**

By: Charles Campbell  
Version 3.0

## Table of Contents

[**Project Software Design Document** 1](#_Toc206171398)

[Table of Contents 2](#_Toc206171399)

[Document Revision History 2](#_Toc206171400)

[Executive Summary 3](#_Toc206171401)

[Design Constraints 3](#_Toc206171402)

[System Architecture View 4](#_Toc206171403)

[Component Diagram 4](#_Toc206171404)

[Sequence Diagram 5](#_Toc206171405)

[Class Diagram 6](#_Toc206171406)

[API Endpoints 7](#_Toc206171407)

[The User Interface 8](#_Toc206171408)

[References 10](#_Toc206171409)

## [Document Revision History](#_heading=h.lnxbz9)

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| 1.0 | 7/18/2025 | Charles Campbell | Completing sections: *Executive Summary, Design Constraints & System Architecture View: Component Diagram* |
| 2.0 | 8/1/2025 | Charles Campbell | Completing sections: *Sequence Diagram, Class Diagram and API endpoints* |
| 3.0 | 8/15/2025 | Charles Campbell | Finishing document; completing sections: |

## [Executive Summary](#_heading=h.35nkun2)

Travlr Getaways requires a well-designed full stack web application for their customers to book travel packages they offer; the users should be able to login, search by location & price, book reservations, and visit the site to review their itineraries for their travels. Additionally, the administrative team should have their own separate page they can exclusively access to maintain the customers/users, as well as make edits or new additions to the trip packages including editing the pricing and a function to either enable or disable packages.

This full stack web development can be done quickly through the MEAN (MongoDB, Express, Angular, Node.js) technologies, a JavaScript based development tool that allows for expedited development of web based applications like Travlr Getaways is looking for. “Applications built with the MEAN stack follow the client-server architecture” (MongoDB, 2025).

MongoDB will help hold the data that will be displayed on the webpages, such as the package names, prices and details for the users; and even be able to hold the data of all the users and package specifics that the admins can view and edit. Node the web server & Express the framework are the foundation utilizing JavaScript to make a quick and easy to maintain code base for the application. Angular is the front end framework that allows for the seamless and scalable user functions, as well as powerful single-page application (SPA) that the administrators are looking for. “The MEAN stack pulls together some of the ‘best-of-breed’ modern web technologies into a powerful, flexible stack” (Holmes & Harber, 2019).

## [Design Constraints](#_heading=h.1ksv4uv)

Should the application become heavy with vast amounts of data, like vast amounts of different packages detailes/prices/descriptions/images or information on users stored in MongoDB, the performance of the database could drop and the overall optimization of data queries and presenting of the data on the application. This could drop user experience and make specific high importance functions for customers frustrating and undesirable.

Angular, despite being a popular framework choice and aims for broad compatibility with web standards in mind, can suffer operating as intended on older versions of web browsers like Internet Explorer 8. Even on up to date versions of browsers, SPAs developed in MEAN might still suffer from slow initial load times due to the upfront loading of the page assets and data from the server.

Finally, MEAN stack is not inherently secure and an application that is not thoroughly tested and resolved of security issues can be prey to the OWASP defined top 10 vulnerabilities that web applications can fall under. “This makes it difficult to ensure complete application protection from hackers and unauthorized users” (Eddie, 2022). Out of date dependencies, improper protection of data either at rest or during client & server transit, and even untested input validation that is ignored can lead to potentially major security breaches and harm both customers and employees/administrators of Travlr Getaways.

## [System Architecture View](#_heading=h.44sinio)

### Component Diagram

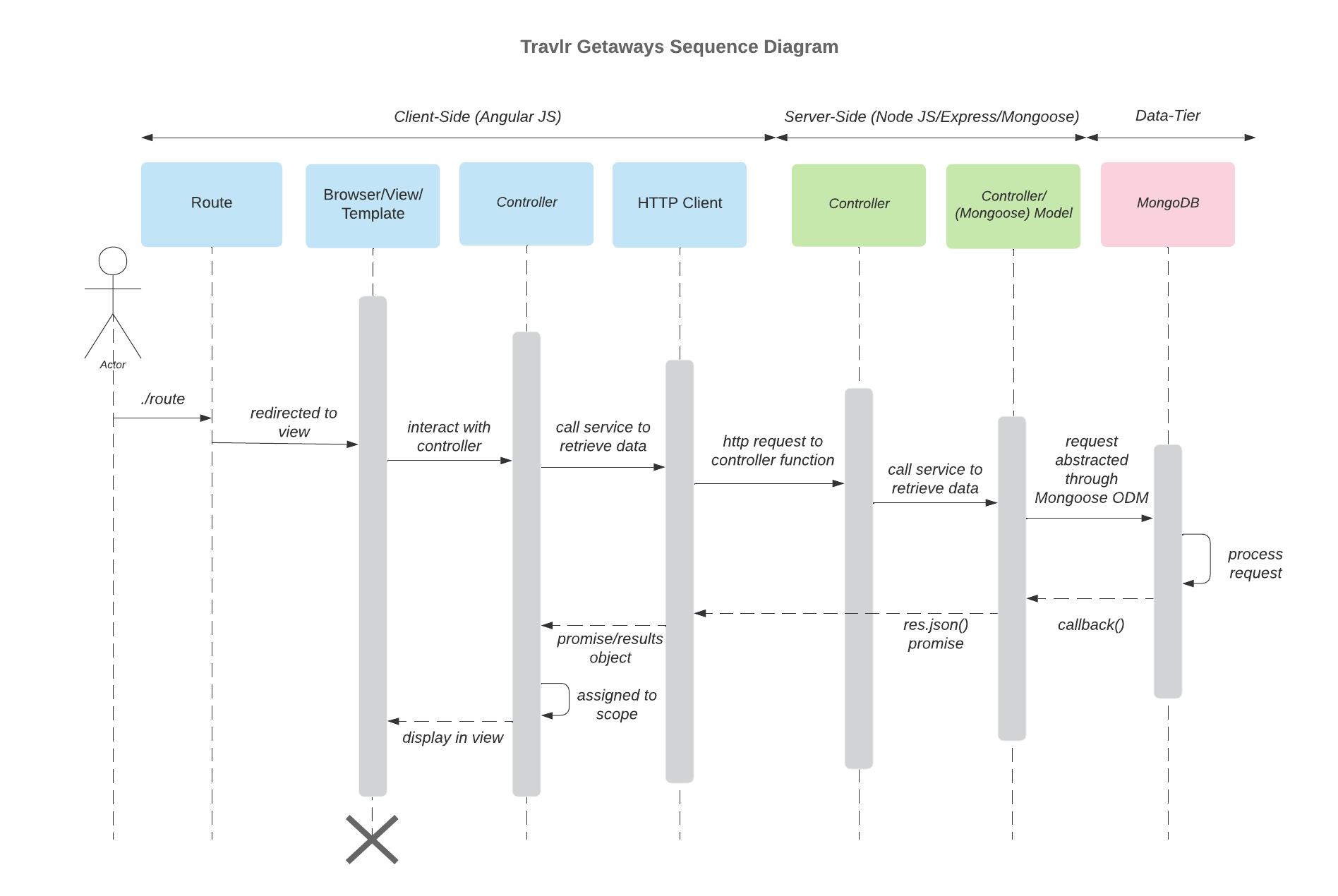


A text version of the component diagram is available: [Travlr Full Stack Component Diagram Text Version](https://learn.snhu.edu/d2l/lor/viewer/view.d2l?ou=6606&loIdentId=24342).

The data held within MongoDB Database can be read by the Mongoose for the server which helps convert the data into a readable format (JSON) and also helps make operation queries easier for the server. This data is also sent to the front-end client. Within the server, utilizing Node.js and Express a server session is made that (through the JavaScript code base) works as an API between the databases to pull and distribute the info in the databases to the front-end, so those with certain authentication parameters can access specific features and operations in the application.

The client side, vastly done with the Angular framework, takes the information stored within MongoDB and utilizes server operations to help display the correct information and gives an interface for users to easily conduct server functions. Each client has their own session that can perform individual server functions, allowing for many users or admins to operate on the data held within the traveler and Mongo databases. For heightened user experience, the graphic library helps display images and appropriate designs that correlate to the items within the databases. This helps make the overall traveler portfolio easy to read and work with. All of this is then displayed on the web browser, so that users and admins can traverse the web application and perform specific actions to accomplish their traveling needs.

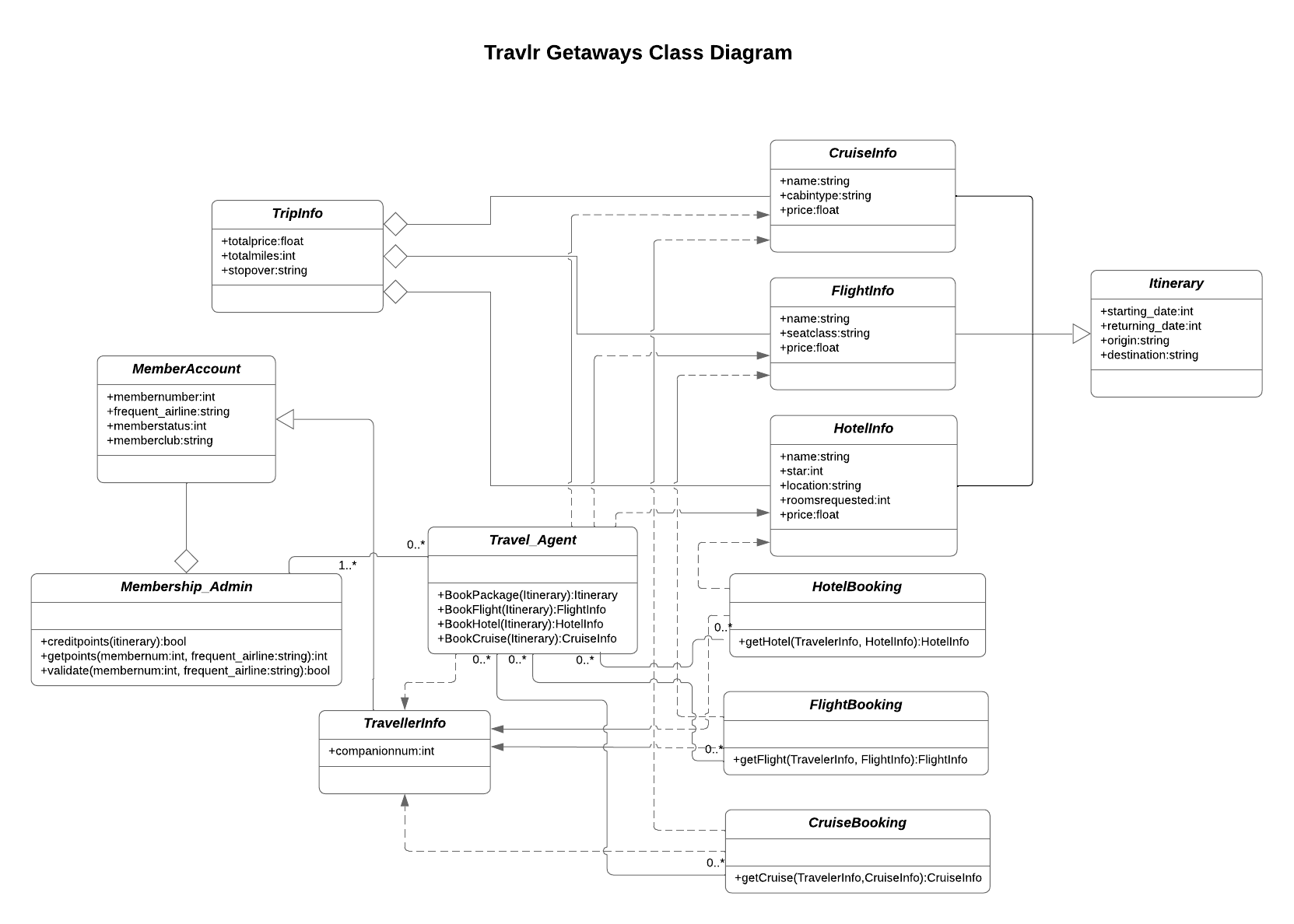
### Sequence Diagram



The actor (or user) accesses a route, this is a defined URL path in the SPA, for example: the admin page where travels can be updated/edited/deleted. The actor utilizing this route allows them to see it on the browser/view. As Holmes & Harber dictate, thanks to Angular’s framework, the router helps the SPA be loaded in the browser which helps navigation and loading for user experience to be much quicker and the only additional loads are API calls sent to the server-side (2019). Should the actor choose to make an action, like deleting a travel package or updating it, this interaction is sent to the controller, which indicates to the HTTP client to connect with server to return the results of a certain API function.

Once the request from the client controller is sent to the server-side Express controller, it then correctly utilizes Mongoose to control that request into the proper format (GET/PUT/etc.) to retrieve that data from the MongoDB. With the actors query properly received, the data is then returned. From the stored BSON in MongoDB, the server side controllers then exposes that data in JSON format. The server returns the results to the client-side HTTP Client & controller. “The Express framework sits on top of Node.js where, through Mongoose, is exposed as JSON” (Holmes & Harber, 2019). With the requested push and pull of data complete, the client controller then displays the results of the correctly processed query to the browser/view. For example, the user deleting a travel package is done seamlessly, thanks to this logical flow, and the results displayed on the browser are almost instantaneous.

### Class Diagram



The classes near the top (above Travel\_Agent) contain data that is important to the travel a user can book. Cruise/Flight/HotelInfo contain important data on the travel specifics that a user needs to view in order to make appropriate choices for their travel plans. These info classes aggregate the variables like price from TripInfo. All of these classes then work together to help establish the itinerary once a user decides upon their travel plans.

Through the Travel\_Agent, a member can then utilize the booking functions for hotel/flight/cruise (which pull the data from the above info classes) to create their travel package. The number of companions on the trip is pulled from the TravellerInfo class as well. The TravellerInfo inherits the member’s data from the MemberAccount which holds important variables like the member number, or the club they’re affiliated with. Additionally, the account aggregates vital functions like validating the members account number, and also either applying additional points based on the data from the generated itinerary, or applying earned points for discounts on their chosen package.

These encapsulated but cohesive classes all work in-tandem in order for the Travel\_Agent to properly help a member book their desired travel pacakage. Not only does it allow the member to correctly create & pay for a solid travel plan (and generate an itinerary from the selected data), but the member can additionally earn or utilize travel points, thus making their overall experience in utilizing Travlr Getaways services more rewarding.

## [API](#_heading=h.2jxsxqh) Endpoints

| **Method** | **Purpose** | **URL** | **Notes** |
| --- | --- | --- | --- |
| **GET** | Retrieve the list of all the trips available | /api/trips | Returns all admin defined available trips for members to take |
| **GET** | Retrieve a single trip from the list | /api/trips/:tripCode | Return a single trip, this method either helps filter the list to a specific trip, or can be used to update/delete from the list |
| **POST** | Add a new trip to the list of available packages | /api/trips | Creates a new trip and adds it to the list of available trips for members |
| **PUT** | Update a trip’s data | /api/trips/:tripCode | Find a single trip within the list, and then allow admin users to update the variables of that trip |
| **DELETE** | Delete a trip from the database | /api/trips/:tripcode | A single trip is found, and then deleted completely from the site |

## 

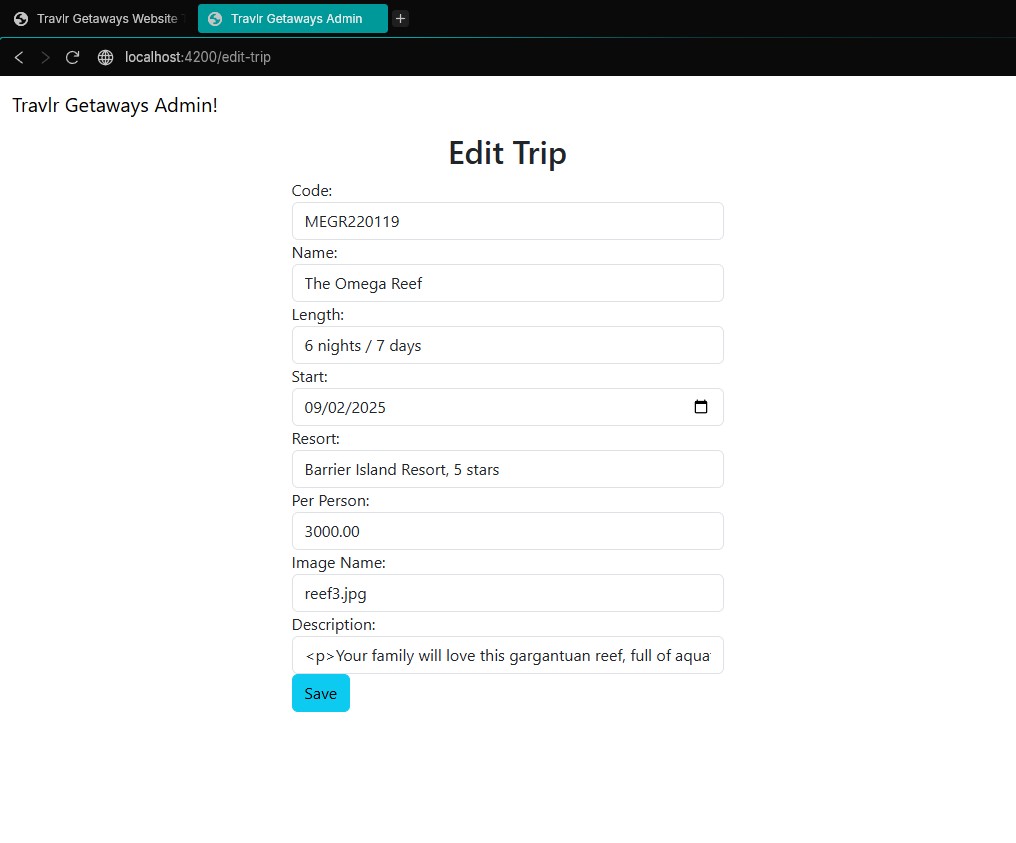
## The User Interface

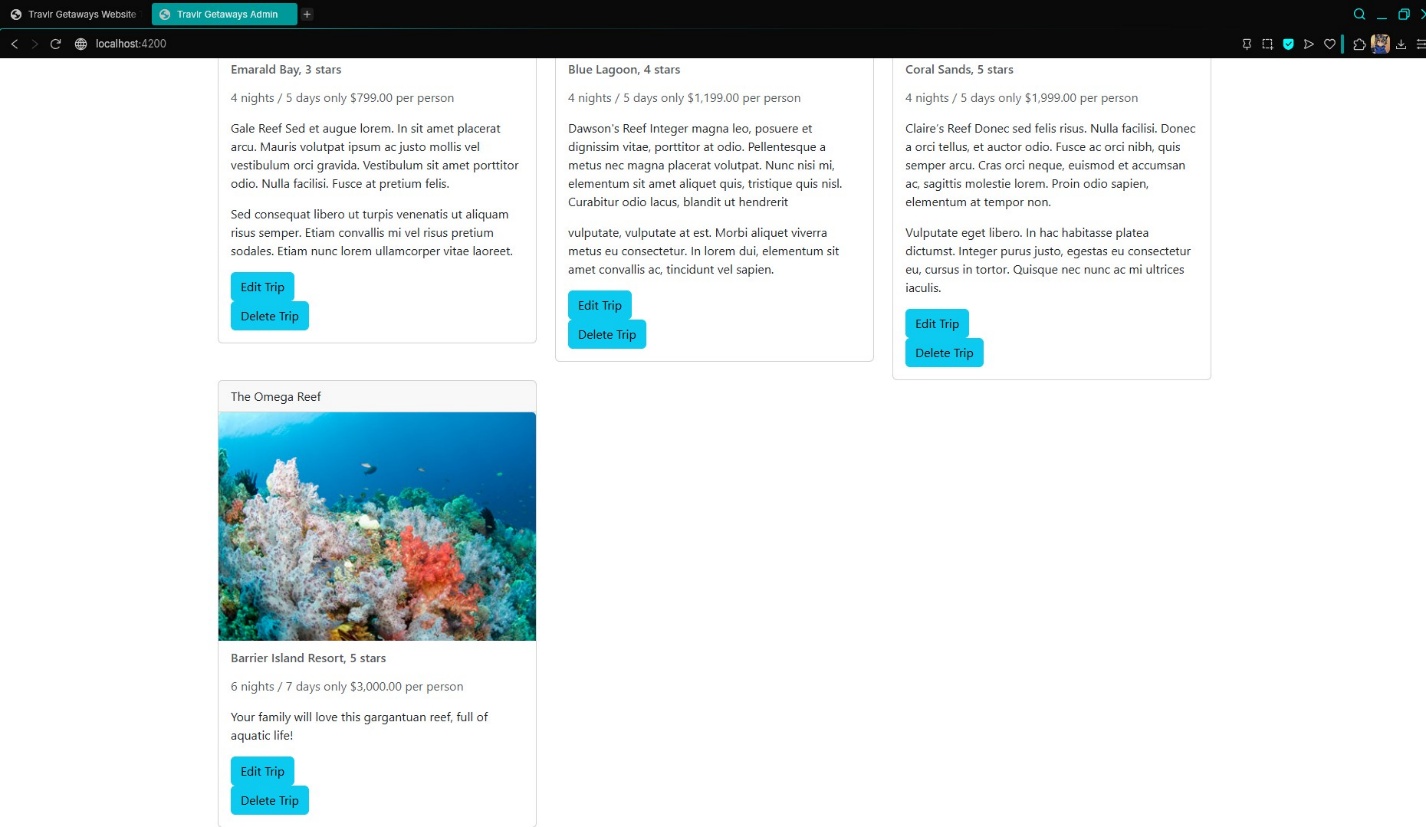
Added trip (Mega Reef):

A screenshot of a computer

AI-generated content may be incorrect.

Edit screen:



Updated List with updated trip:  
  


The Angular project’s structure was greatly improved thanks to the ability to generate components using the Angular CLI. This component-based architecture made the development and testing of the project much better than the Express user functions and methods. Not only does it allow the use of Separation of Concerns (SoC) which helped with code organization, easier debugging, and increased modularity, it allowed the Angular admin site to be an easier development process than the Express counterpart.

With the help of the Angular framework, the overall functionality provided is greatly enhanced as well by comparison to the Express user front-end. For example, only the use of reading data from the MongoDB is used in the Express site, where it simply displays the data to the users on the travel page. In the richer Angular developed admin site, users can utilize full CRUD methods. Administrators can not only read and update current listed trips, but they can also create new additions that will be archived directly into the database, along with having the option to delete any should they need to.

Another benefit that Angular provides is the robust ability to test and improve the code. In part thanks to the use of the components and Separation of Concerns, the Angular admin site was much easier to identify bugs and quickly refactor and allow proper implementation of the API’s get & put functions. And rather than just the system terminal that might provide simple error catches a developer puts in the code itself, Angulars built-in ErrorHandler class helps developers see descriptive errors in the browser console. This helped tremendously in the testing and debugging of adding the security feature, the user authentication (login), to ensure that the API get & put features only worked when an administrative user was logged into the site. Thus, the data within the database not only could operate normally in the SPA thanks to the enhanced Angular testing, but also the security of the data was ensured as well.

## References

Eddie, J. (2022, January 7). *Why MEAN Stack is Best for Web Development?* Medium. <https://enlear.academy/why-mean-stack-is-best-for-web-development-55b7e4a00d72>

Holmes, S., & Harber, C.. (2019). *Getting MEAN with Mongo, Express, Angular, and Node*. Manning. <https://learning.oreilly.com/library/view/getting-mean-with/9781617294754>

*How To Use The MEAN Stack: Build A Web Application From Scratch*. (2025). MongoDB. <https://www.mongodb.com/resources/languages/mean-stack-tutorial#what-is-the-mean-stack>